User's Manual

RZ/A2M グループ

DRP Driver ユーザーズマニュアル

本資料に記載の全ての情報は本資料発行時点のものであり、ルネサス エレクトロニクスは、
予告なしに、本資料に記載した製品または仕様を変更することがあります。
ルネサス エレクトロニクスのホームページなどにより公開される最新情報をご確認ください。

ルネサス エレクトロニクス
www.renesas.com

Rev.1.01 2019.05

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。 たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。 なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

このマニュアルの使い方

1. 目的と対象者

このマニュアルはソフトウェア「DRP Driver」の機能、使用方法をユーザーに理解していただくためのマニュアルです。本ソフトウェアを用いた応用システムを設計するユーザーを対象にしています。このマニュアルを使用するには、プログラミング言語、マイクロコンピュータに関する基本的な知識が必要です。

本ソフトウェアは、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。

目次

1. はじめに

1.1 要旨

本書は RZ/A2M グループのマイクロコンピュータに搭載されている DRP(Dynamic Reconfigurable Processor)を制御するソフトウェア「DRP Driver」の機能、使い方について説明します。

1.2 機能

DRP はユーザーの設定に応じて、様々な機能を実現することができます。本書では、DRP で実現された機能を「回路」と呼び、回路情報を表すデータを「コンフィグレーションデータ※」と呼びます。コンフィグレーションデータの実体はメモリ上に配置されたバイナリデータです。

DRP Driver は DRP のデバイスドライバとして、以下の機能を持ちます。

- DRP へクロックを供給、DRP Driver を初期化する
- DRP のクロックを停止、DRP Driver を終了する
- DRP へコンフィグレーションデータをロードする
- DRP へロードされたコンフィグレーションデータを消去する（本書ではアンロードと呼びます）
- DRP へ書き込まれた回路へクロックを供給、有効化する
- DRP へ書き込まれた回路のクロックを停止、無効化する
- DRP へ書き込まれた回路へ動作パラメータを設定、動作を開始する
- DRP へ書き込まれた回路の動作完了を通知する
- DRP へ書き込まれた回路の状態（有効か無効か、動作中か否尾か、など）を取得する
- メモリ上のコンフィグレーションデータの情報（バージョンなど）を取得する
- メモリ上のコンフィグレーションデータの CRC チェック

※ コンフィグレーションデータは DRP Library として提供されます。DRP Library についての詳細は、「RZ/A2M グループ DRP Library ユーザーズマニュアル」（R01US0367）を参照してください。

1.3　ソフトウェア構成

DRP Driver のソフトウェア構成を以下に示します。DRP Driver はインターフェース部分とコア部分から構成され、それぞれ、ソースコードで提供されます。DRP Driver は OS abstraction layer を介して、FreeRTOS に対応します。

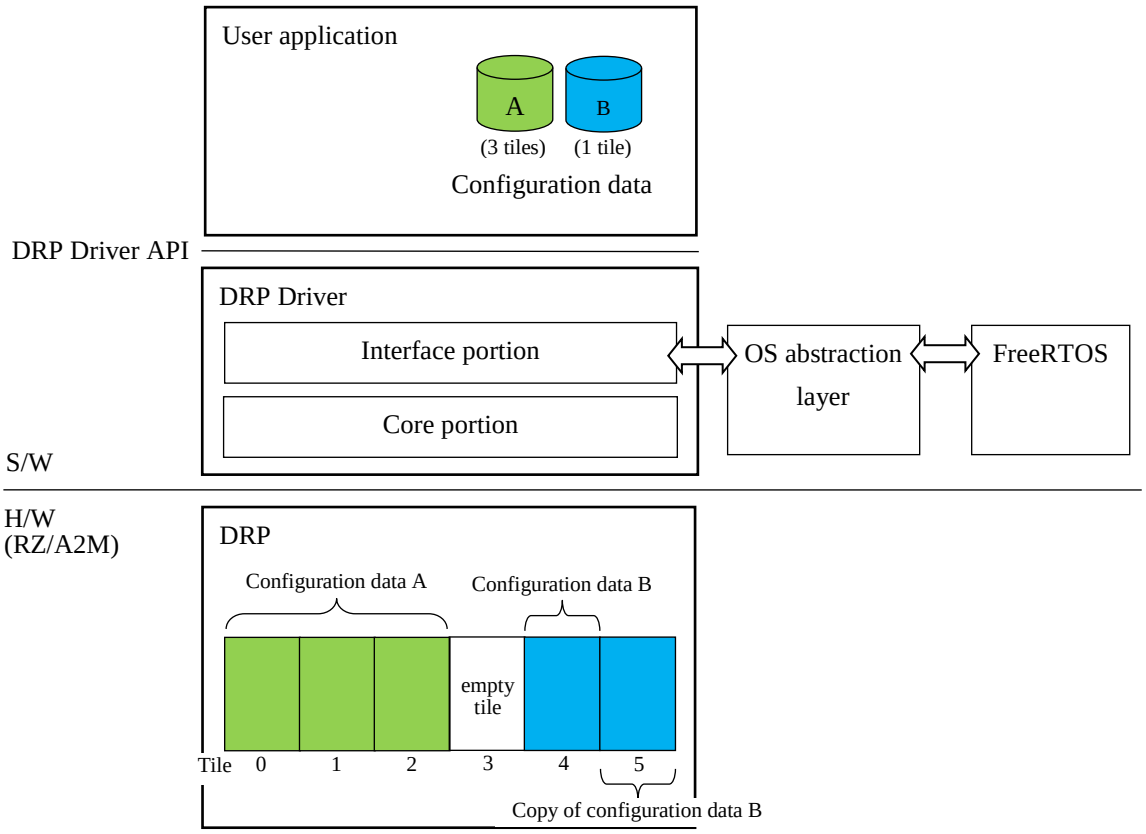


図1.1　ソフトウェア構成

- DRP はコンフィグレーションデータをロードするための 6 個の「タイル」と呼ばれる領域を持ちます。
- コンフィグレーションデータはタイル単位でロードされます。
- コンフィグレーションデータは 1 から 6 までの整数であらわされる固有のタイル数を持ちます。タイル数は占有するタイルの数を表します。
- タイル数が 3 以下のコンフィグレーションデータは同じもの複数ロードすることができます。
- 本書では、DRP の持つ 6 つのタイルをそれぞれ、タイル 0 からタイル 5 と呼称します。
- 上記の図は、タイル数 3 のコンフィグレーションデータ A をタイル 0 からタイル 2 に一つ配置、タイル数 1 のコンフィグレーションデータ B をタイル 4 とタイル 5 に二つ配置した例を表します。

2. 動作条件

DRP Driver は下記の条件で動作します。

表2.1　動作条件

項目	内容
マイクロコンピュータ	RZ/A2M グループに属するマイクロコンピュータ搭載の Cortex™-A9 上で動作 対応する RZ/A2M グループのマイクロコンピュータの型名は下記となります※： - R7S921051VCBG - R7S921052VCBG - R7S921053VCBG
開発環境	e2 studio V7.1.0 対応するツールチェーンは下記となります： GNU Arm Embedded Toolchain 6-2017-q2-update

※　DRP Driver は DRP 機能を搭載した RZ/A2M グループに属するマイクロコンピュータで動作します。DRP 機能を搭載していない RZ/A2M グループに属するマイクロコンピュータでは動作しませんのでご注意ください。

3. ファイル構成

DRP Driver のファイル構成を図3.1に示します。

```
src
    renesas
        drivers
            drp
                inc
                    r_dk2_if.h          Header file of DRP Driver interface part
                src
                    drp_iodefine.h      IO definition file of DRP
                    r_dk2_core.c        Source file of DRP Driver core part
                    r_dk2_core.h        Header file of DRP Driver core part
                    r_dk2_if.c          Source file of DRP Driver interface part
```

図3.1 ファイル構成

4. API 仕様

4.1 API 関数一覧

表4.1に DRP Driver の API 関数の一覧を示します。

表4.1 DRP Driver の API 関数一覧

API 関数名	概要	ページ
R_DK2_Initialize	DRP Driver の初期化、及び、DRP の初期化	7
R_DK2_Uninitialize	DRP の停止、及び、DRP Driver の終了	8
R_DK2_Load	コンフィグレーションデータを DRP へロード	9
R_DK2_Unload	コンフィグレーションデータを DRP からアンロード	14
R_DK2_Activate	DRP 上の回路の有効化	15
R_DK2_Inactivate	DRP 上の回路の無効化	16
R_DK2_Start	DRP 上の回路の動作を開始	17
R_DK2_GetStatus	DRP 上の回路の状態を取得	19
R_DK2_GetInfo	コンフィグレーションデータの情報を取得、CRC のチェック	20
R_DK2_GetVersion	DRP Driver のバージョン情報を取得	21

全ての API 関数は割込みコンテキストからコールすることはできません。API 関数のリエントラント可能性については「8 OS依存部」を参照してください。

4.2 エラーコード

DRP Driver の API 関数は戻り値が 0、または、正の数のとき、正常終了したことを表し、戻り値が負の数のとき、異常終了したことを表します。異常終了時にはエラーコードを返却します。表4.2にエラーコードの一覧を示します。エラーが発生する詳細な条件は、「5 APIリファレンス」の各 API 関数の戻り値の説明を参照してください。

表4.2 関数のエラーコード一覧

マクロ名	値	内容
R_DK2_SUCCESS	0	正常終了
R_DK2_ERR_ARG	-1	引数エラー
R_DK2_ERR_FORMAT	-2	フォーマットエラー
R_DK2_ERR_CRC	-3	CRC エラー
R_DK2_ERR_DEVICE	-4	デバイスエラー
R_DK2_ERR_BUSY	-5	ビジー
R_DK2_ERR_INTERNAL	-6	内部エラー
R_DK2_ERR_OVERWRITE	-7	データ上書きエラー
R_DK2_ERR_OS	-8	OS エラー
R_DK2_ERR_STATUS	-9	状態エラー
R_DK2_ERR_TILE_PATTERN	-10	タイルパターンエラー
R_DK2_ERR_STOPPED	-11	転送停止エラー

5.　API リファレンス

5.1　　API リファレンスの読み方

API 関数名	分類
機能概要	同期/非同期関数
書式	API の呼出し形式を示します。#include “ヘッダファイル”で示すヘッダファイルは、この API の実行に必要な標準ヘッダファイルです。必ずインクルードしてください。 I,O は、引数がそれぞれ入力データ、出力データであることを意味します。IO の場合は入出力データであることを意味します。
戻り値	API の戻り値を示します。戻り値の後に「：」を付けて記載されているコメントは、その戻り値についての説明（リターン条件等）です。
解説	API の仕様について説明します。
注意	注意事項があればここに示します。

5.2 R_DK2_Initialize

R_DK2_Initialize			DRP Driver API
DRP Driver の初期化、及び、DRP の初期化			同期関数
書式	#include "r_dk2_if.h" int32_t R_DK2_Initialize(void);		
戻り値	R_DK2_SUCCESS	:	正常終了。
	R_DK2_ERR_DEVICE	:	異常終了。 DRP の初期化に失敗した場合、本エラーが発生します。
	R_DK2_ERR_OS	:	異常終了。 OS 資源の確保に失敗した場合、本エラーが発生します。
	R_DK2_ERR_STATUS	:	異常終了。 DRP Driver が既に初期化されている場合、本エラーが発生します。
解説	本 API 関数は、内部変数を初期化、OS 資源を確保して、DRP Driver を使用できる状態にします。 また、DRP を低消費電力モードから復帰してクロック供給を開始、ハードウェアの初期化を行います。		
注意	R_DK2_ERR_DEVICE が発生した場合は、ご使用のデバイスを確認してください。DRP Driver は、DRP 機能を搭載した RZ/A2M グループに属するマイクロコンピュータに対応しています。DRP Driver の詳しい動作条件は、「2 動作条件」を参照してください。 戻り値が R_DK2_ERR_OS となった場合は、OS の設定を見直してください。		

5.3 R_DK2_Uninitialize

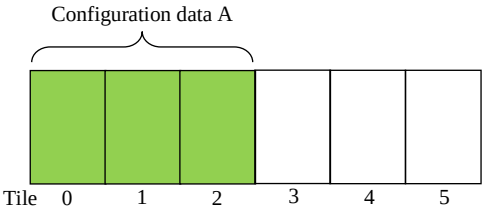
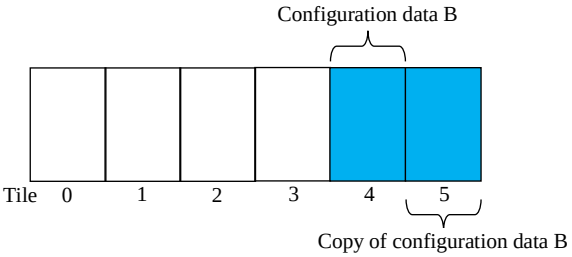
R_DK2_Uninitialize			DRP Driver API
DRP の停止、及び、DRP Driver の終了			同期関数
書式	`#include "r_dk2_if.h"` `int32_t R_DK2_Uninitalize(void);`		
戻り値	R_DK2_SUCCESS	:	正常終了。
	R_DK2_ERR_OS	:	異常終了。 OS 資源の開放に失敗した場合、本エラーが発生します。
	R_DK2_ERR_STATUS	:	異常終了。 DRP Driver が既に終了している場合、本エラーが発生します。
解説	本 API 関数は、DRP へのクロック供給を停止し、DRP を低消費電力モードへ移行します。DRP が動作中の場合でも強制停止します。 また、OS 資源を開放し、DRP Driver の状態を未初期化状態へ移行します。本 API 関数実行後は、再度、R_DK2_Initialize 関数をコールするまで、DRP Driver は使用できない状態になります。		
注意	本 API 関数は DRP が動作中の場合でも強制的に停止します。その場合、R_DK2_Load 関数で設定したコールバック関数がコールされない可能性がありますので、ご注意ください。 戻り値が R_DK2_ERR_OS となった場合は、OS の設定を見直してください。		

5.4 R_DK2_Load

R_DK2_Load	DRP Driver API
コンフィグレーションデータを DRP へロード	同期・非同期関数

書式

```
#include "r_dk2_if.h"
int32_t R_DK2_Load(const void *const pconfig, const uint8_t top_tiles, const
uint32_t tile_pattern, const load_comp_t pload, const process_comp_t pprocess,
uint8_t *const paid);
```

pconfig	I	ロードするコンフィグレーションデータのアドレスを指定してください。 コンフィグレーションデータは 32 バイト境界に整列させてください。また、コンフィグレーションデータは、物理メモリ上に存在する必要があります。
top_tiles	I	DRP が持つ、タイル 0 からタイル 5 までの 6 つのタイルを表すマクロ「R_DK2_TILE_0」～「R_DK2_TILE_5」を使用して、コンフィグレーションデータを配置する先頭タイル位置を指定します。 コンフィグレーションデータを複数ロードする場合は、上記マクロのビット毎の論理和をとってください。 例えば、タイル数 3 のコンフィグレーションデータ A をタイル 0 からタイル 2 へ一つ配置する場合は、「R_DK2_TILE0」と指定してください。  例えば、タイル数 1 のコンフィグレーションデータ B をタイル 4 とタイル 5 に二つ配置する場合は、 「R_DK2_TILE_4 | R_DK2_TILE_5」と指定してください。 
tile_pattern	I	タイルパターンを指定します。設定値については、「5.4.1 タイルパターン」を参照してください。 タイルパターンは、一度、設定したら、R_DK2_Unload 関数で全タイルのコンフィグレーションデータをアンロードするまで同じ設定を使用してください。 既にコンフィグレーションデータをロードした状態で、タイルパターンを変更した場合は、本 API 関数は戻り値 R_DK2_ERR_TILE_PATTERN を返して失敗します。

	pload	I	コンフィグレーションデータのロード完了を通知するコールバック関数のアドレスを指定してください。 引数 pload で指定するコールバック関数の詳しい仕様は「5.4.2 ロード完了コールバック関数」を参照してください。 本引数に NULL 以外を指定した場合は、R_DK2_Unload 関数でコンフィグレーションデータのロードを中断可能です。本引数に NULL を指定した場合は、R_DK2_Unload 関数でコンフィグレーションデータのロードを中断することはできず、本 API 関数は、ロードが完了してから終了します。
	pprocess	I	R_DK2_Start 関数で開始した処理の完了を通知するコールバック関数のアドレスを指定してください。 引数 pprocess で指定するコールバック関数の詳しい仕様は「5.8.1 処理完了コールバック関数」を参照してください。 NULL を指定した場合は、この通知は行われません。
	paid	O	ロードしたコンフィグレーションデータを識別する ID を通知するため、ユーザー側で用意した要素数 6 の配列のアドレスを指定してください。 配列の添え字 0 から添え字 5 は、DRP の持つ、タイル 0 からタイル 5 までの 6 つのタイルを表し、配列の要素は該当タイルにロードされたコンフィグレーションデータの ID を表します。 複数タイルのコンフィグレーションデータの場合は、該当する全てのタイルを表す配列の要素に同じ ID が格納されます。この ID は回路ごとに一意の正の数で、0 はコンフィグレーションデータがロードされていないことを意味します。 同じコンフィグレーションデータが複数、配置されている場合は、それぞれ異なる ID が割り振られます。 本引数で通知される ID は、今までに書き込んだ全てのコンフィグレーションデータを含めて、R_DK2_Load 関数実行後の ID が 6 タイル分通知されます。 NULL を指定した場合は、この通知は行われません。 例えば、タイル数 3 のコンフィグレーションデータ A をタイル 0 からタイル 2 へ一つ配置、タイル数 1 のコンフィグレーションデータ B をタイル 4 とタイル 5 に二つ配置した場合は、以下のような配列となります。

添え字	内容
0	コンフィグレーションデータ A の回路情報をもつ回路の ID
1	添え字 0 と同じ
2	添え字 0 と同じ
3	0
4	コンフィグレーションデータ B の回路情報を持つ回路の ID
5	コンフィグレーションデータ B の回路情報を持つ回路の ID（添え字 4 とは異なる）

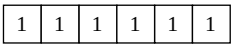
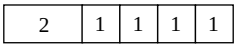
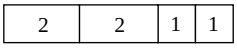
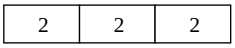
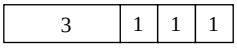
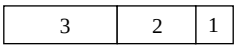
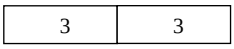
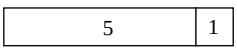
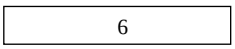
戻り値	R_DK2_SUCCESS	:	正常終了。
	R_DK2_ERR_ARG	:	異常終了。 以下の場合、本エラーが発生します。 - 引数 pconfig に NULL が指定された - 引数 pconfig に 32 バイト境界に整列していない値が指定された - 引数 top_tiles が「R_DK2_TILE_0」～「R_DK2_TILE_5」のビット毎の論理和の形になっていない - 引数 tile_pattern に表5.1に示すマクロ以外が指定された
	R_DK2_ERR_FORMAT	:	異常終了。 コンフィグレーションデータのフォーマット不整合を検出した場合、本エラーが発生します。

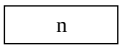
	R_DK2_ERR_DEVICE	:	異常終了。 引数 pload に NULL を指定された、かつ、コンフィグレーションデータのロード時に転送エラーが発生した場合に、本エラーが発生します。
	R_DK2_ERR_BUSY	:	異常終了。 引数 pload に NULL 以外を設定し、コンフィグレーションデータのロード中に別のコンフィグレーションデータをロードしようとした場合、本エラーが発生します。
	R_DK2_ERR_OVERWRITE	:	異常終了。 指定されたコンフィグレーションデータのロード位置に既に他のコンフィグレーションデータが書き込まれている場合、本エラーが発生します。
	R_DK2_ERR_OS	:	異常終了。 OS による排他制御に失敗した場合、本エラーが発生します。
	R_DK2_ERR_STATUS	:	異常終了。 DRP Driver が初期化されていない場合、本エラーが発生します。
	R_DK2_ERR_TILE_PATTERN	:	異常終了。 下記の場合、本エラーが発生します。 - 既にコンフィグレーションデータがロードされている状態でタイルパターンが変更された - タイルパターンに合致しないタイル位置、タイル数のコンフィグレーションデータが指定された
解説	本 API 関数は、引数 pload に NULL 以外を指定した場合は、指定されたコンフィグレーションデータを DRP へロード開始し、ロードの完了はコールバック関数により通知されます。このとき、ロード完了前に、他のコンフィグレーションデータをロードはできません。その場合、戻り値 R_DK2_ERR_BUSY を返却して、本 API 関数は失敗します。また、引数 pload に NULL 以外を指定した場合は、R_DK2_Unload 関数でコンフィグレーションデータのロードを中断することができます。 引数 pload に NULL を指定した場合は、本 API 関数実行時にコンフィグレーションデータのロードを完了します。このとき、R_DK2_Unload 関数でコンフィグレーションデータのロードを中断することはできません。 また、本 API 関数は、指定されたコンフィグレーションデータを複数のタイル位置にロードすることもできます。 引数 pload で指定するコールバック関数についての詳細は「5.4.2 ロード完了コールバック関数」、引数 pprocess で指定するコールバック関数についての詳細は「5.8.1 処理完了コールバック関数」を参照してください。 本 API 関数は、複数の DRP Driver の API 関数が同時に実行されないように OS の機能で排他制御されています。排他制御時の資源獲得がタイムアウトなどで失敗した場合には、戻り値 R_DK2_ERR_OS を返して本 API 関数は失敗します。		
注意	戻り値が R_DK2_ERR_FORMAT となった場合は、引数 pconfig で指定したアドレスが、正しいコンフィグレーションデータのアドレスとなっているか確認してください。 戻り値が R_DK2_ERR_DEVICE となった場合は、コンフィグレーションデータの転送中にエラーが発生したことを示します。コンフィグレーションデータを配置したメモリの設定などを見直してください。 引数 pconfig で設定するコンフィグレーションデータが Cortex-A9 のキャッシュ上などに存在し、物理メモリの内容がコンフィグレーションデータと一致しない状態になっていると、正常にロードできません。本 API 関数コール前にキャッシュのクリーンを行う、または、コンフィグレーションデータを非キャッシュ領域に配置するなどの対処を行う必要があります。		

5.4.1 タイルパターン

コンフィグレーションデータを DRP へロードするときのタイル数とタイル位置の組み合わせは、表5.1に示す 11 パターンに限られます。使用する組み合わせに合わせて、下記のマクロ値を R_DK2_Load 関数の引数 tile_pattern に設定してください。

表5.1　タイルパターン一覧

タイルパターン	R_DK2_Load 関数の引数 tile_pattern に設定するマクロ
	R_DK2_TILE_PATTERN_1_1_1_1_1_1
	R_DK2_TILE_PATTERN_2_1_1_1_1
	R_DK2_TILE_PATTERN_2_2_1_1
	R_DK2_TILE_PATTERN_2_2_2
	R_DK2_TILE_PATTERN_3_1_1_1
	R_DK2_TILE_PATTERN_3_2_1
	R_DK2_TILE_PATTERN_3_3
	R_DK2_TILE_PATTERN_4_1_1
	R_DK2_TILE_PATTERN_4_2
	R_DK2_TILE_PATTERN_5_1
	R_DK2_TILE_PATTERN_6


：タイル数nのコンフィグレーションデータ

5.4.2 ロード完了コールバック関数

ロード完了コールバック関数

callback function

コンフィグレーションデータのロード完了 同期関数

書式	#include "r_dk2_if.h" void load_comp(uint8_t id, int32_t result); ※ 関数名は自由に命名できます。		
	id	I	ロード完了した回路の ID
	result	I	R_DK2_SUCCESS : ロードが正常に完了したことを表します。 R_DK2_ERR_DEVICE : コンフィグレーションデータのロード時に転送エラーが発生したことを表します。 R_DK2_ERR_STOPPED : R_DK2_Unload 関数コールにより、コンフィグレーションデータのロード時に転送が停止したことを表します。
戻り値	なし		
解説	R_DK2_Load 関数の引数 pload で指定するコールバック関数です。コンフィグレーションデータのロード完了を通知します。コンフィグレーションデータを複数ロードした場合は、ロードした個数分、本コールバック関数がコールされます。 本関数は割込みコンテキストで実行されます。本関数内で、DRP Driver の関数はコールしないでください。		
注意	引数 result が R_DK2_ERR_DEVICE のときは、コンフィグレーションデータを配置したメモリの設定などを見直してください。		

5.5 R_DK2_Unload

R_DK2_Unload			DRP Driver API
コンフィグレーションデータを DRP からアンロード			同期関数
書式	#include "r_dk2_if.h" int32_t R_DK2_Unload(const uint8_t id, uint8_t *const paid);		
	id	I	アンロードする回路の ID を指定してください。複数の回路をアンロードする場合は、それぞれの回路の ID のビット毎の論理和を指定してください。 0 を指定した場合は、ロードされている全ての回路をアンロードします。
	paid	O	本 API 関数実行後の DRP のロード状況を通知するため、ユーザー側で用意した要素数 6 の配列のアドレスを指定してください。 配列の添え字 0 から添え字 5 は、DRP の持つ、タイル 0 からタイル 5 までの 6 つのタイルを表し、配列の要素は該当タイルにロードされたコンフィグレーションデータの ID を表します。 複数タイルのコンフィグレーションデータの場合は、該当する全ての配列の要素に同じ ID が格納されます。 この ID は回路ごとに一意の正の数で、0 はコンフィグレーションデータがロードされていないことを意味します。 同じコンフィグレーションデータが複数、配置されている場合は、それぞれ異なる ID が割り振られます。 本引数で通知される ID は、今までに書き込んだ全てのコンフィグレーションデータを含めて、R_DK2_Unload 関数実行後の ID が 6 タイル分通知されます。 NULL を指定した場合は、この通知は行われません。
戻り値	R_DK2_SUCCESS	:	正常終了。
	R_DK2_ERR_ARG	:	異常終了。 引数 id が DRP 上にロードされた回路ではない場合、本エラーが発生します。
	R_DK2_ERR_OS	:	異常終了。 OS による排他制御に失敗した場合、本エラーが発生します。
	R_DK2_ERR_STATUS	:	異常終了。 下記の場合、本エラーが発生します。 - DRP Driver が初期化されていない
解説	本 API 関数は、指定された ID の回路を DRP からアンロードします。回路をアンロードすることにより、同じタイル位置に再度、コンフィグレーションデータをロードすることができるようになります。回路のロード中の場合、または、回路が動作中の場合でも、本 API 関数で強制的にアンロードします。 コンフィグレーションデータのロード時に本 API 関数がコールされ、転送が停止した場合、R_DK2 複数の回路、または、ロードされた全ての回路をアンロードすることもできます。 本 API 関数は、複数の DRP Driver の API 関数が同時に実行されないように OS の機能で排他制御されています。排他制御時の資源獲得がタイムアウトなどで失敗した場合には、戻り値 R_DK2_ERR_OS を返して本関数は失敗します。		
注意	なし。		

5.6 R_DK2_Activate

R_DK2_Activate			DRP Driver API
DRP 上の回路の有効化			同期関数
書式	#include "r_dk2_if.h" int32_t R_DK2_Activate(const uint8_t id, const uint32_t freq);		
	id	I	有効化する回路の ID を指定して下さい。複数の回路を有効化する場合は、それぞれの回路の ID のビット毎の論理和を指定して下さい。 0 を指定した場合は、ロードされている全ての回路を有効化します。
	freq	I	0 を設定してください。
戻り値	R_DK2_SUCCESS	:	正常終了。
	R_DK2_ERR_ARG	:	異常終了。 引数 id が DRP 上に書き込まれた回路の物ではない場合、本エラーが発生します。
	R_DK2_ERR_OS	:	異常終了。 OS による排他制御に失敗した場合、本エラーが発生します。
	R_DK2_ERR_STATUS	:	異常終了。 下記の場合、本エラーが発生します。 - DRP Driver が初期化されていない - 引数 id で指定した回路が Loaded state でない - 引数 id に 0 を指定した、かつ、Loaded state の回路が存在しない （回路の状態については、「6.2 回路毎の状態遷移」参照）
解説	本 API 関数は DRP 上にロードされた回路を有効化して、該当タイルにクロックを供給、回路として使用できる状態にします。 複数の回路、または、ロードされた全ての回路を有効化することもできます。引数 id に 0 を指定して全ての回路を有効化する場合は、Loaded state の回路のみが対象となります。（回路の状態については、「6.2 回路毎の状態遷移」参照） 本 API 関数は、複数の DRP Driver の API 関数が同時に実行されないように OS の機能で排他制御されています。排他制御時の資源獲得がタイムアウトなどで失敗した場合には、戻り値 R_DK2_ERR_OS を返して本関数は失敗します。		
注意	なし。		

5.7 R_DK2_Inactivate

R_DK2_Inactivate			DRP Driver API
DRP 上の回路の無効化			同期関数
書式	#include "r_dk2_if.h" int32_t R_DK2_Inactivate(const uint8_t id);		
	id	I	無効化する回路の ID を指定してください。複数の回路を無効化する場合は、それぞれの回路の ID のビット毎の論理和を指定してください。 0 を指定した場合は、ロードされている全ての回路を無効化します。
戻り値	R_DK2_SUCCESS	:	正常終了。
	R_DK2_ERR_ARG	:	異常終了。 引数 id が DRP 上に書き込まれた回路のものではない場合、本エラーが発生します。
	R_DK2_ERR_OS	:	異常終了。 OS による排他制御に失敗した場合、本エラーが発生します。
	R_DK2_ERR_STATUS	:	異常終了。 下記の場合、本エラーが発生します。 - DRP Driver が初期化されていない - 引数 id で指定した回路が Activated state、Started state ではない - 引数 id に 0 を指定した、かつ、Activated state、Started state の回路が存在しない （回路の状態については、「6.2 回路毎の状態遷移」参照）
解説	本 API 関数は DRP 上にロードされた回路を無効化して、該当タイルのクロックを停止、低消費電力状態にします。回路が動作中の場合でも強制的に無効化します。 複数の回路、または、ロードされた全ての回路を無効化することもできます。引数 id に 0 を指定して全ての回路を無効化する場合は、Activated state、及び、Started state の回路のみが対象になります。 本 API 関数は、複数の DRP Driver の API 関数が同時に実行されないように OS の機能で排他制御されています。排他制御時の資源獲得がタイムアウトなどで失敗した場合には、戻り値 R_DK2_ERR_OS を返して本関数は失敗します。		
注意	なし。		

5.8 R_DK2_Start

R_DK2_Start			DRP Driver API
DRP 上の回路の動作を開始			非同期関数
書式	#include "r_dk2_if.h" int32_t R_DK2_Start(const uint8_t id, const void *const pparam, const uint32_t size);		
	id	I	動作を開始する回路の ID を指定してください。
	pparam	I	回路の動作のためのパラメータを格納した領域を指定してください。パラメータを格納した領域は、物理メモリ上に存在する必要があります。パラメータを格納した領域は、各回路が独立に読出しを行うため、複数の回路で共有しないでください。 パラメータの仕様はコンフィグレーションデータ毎に異なります。各コンフィグレーションデータのパラメータ仕様は「RZ/A2M グループ DRP Library ユーザーズマニュアル」（R01US0367）を参照してください。
	size	I	引数 pparam で指定したパラメータ領域のサイズを指定してください。
戻り値	R_DK2_SUCCESS	:	正常終了。
	R_DK2_ERR_ARG	:	異常終了。 下記の場合、本エラーが発生します。 - 引数 id が DRP 上にロードされた回路ではない - 引数 pparam に NULL が指定された - 引数 size に 0 が指定された
	R_DK2_ERR_OS	:	異常終了。 OS による排他制御に失敗した場合、本エラーが発生します。
	R_DK2_ERR_STATUS	:	異常終了。 下記の場合、本エラーが発生します。 - DRP Driver が初期化されていない - 引数 id で指定した回路が Activated state ではない （回路の状態については、「6.2 回路毎の状態遷移」参照）
解説	本 API 関数は DRP 上の回路の動作を開始します。処理の完了は、R_DK2_Load 関数の引数 pprocess で指定した処理完了コールバック関数で通知されます。 処理完了コールバック関数の詳細は「5.8.1 処理完了コールバック関数」を参照してください。 本 API 関数は、複数の DRP Driver の API 関数が同時に実行されないように OS の機能で排他制御されています。排他制御時の資源獲得がタイムアウトなどで失敗した場合には、戻り値 R_DK2_ERR_OS を返して本関数は失敗します。		
注意	引数 pparam で設定するパラメータを格納する領域、及び、回路の入出力データが Cortex-A9 のキャッシュ上などに存在し、物理メモリの内容がパラメータ、及び、回路の入出力データと一致しない状態になっていると、顔路が正常に動作しません。本 API 関数コール前にキャッシュのクリーンを行う、または、パラメータ、及び、回路の入出力データを非キャッシュ領域に配置するなどの対処を行う必要があります。		

5.8.1 処理完了コールバック関数

処理完了コールバック関数

callback function

R_DK2_Start で開始した処理の完了

同期関数

書式

```
#include "r_dk2_if.h"
void process_comp(uint8_t id, int32_t result);
```

※ 関数名は自由に命名できます。

id	I	処理完了した回路の ID
result	I	R_DK2_SUCCESS ： 処理が正常に完了したことを表します。 R_DK2_ERR_DEVICE ： R_DK2_Start 関数で設定したパラメータの転送時、または、回路の入出力データの転送時に転送エラーが発生したことを表します。 R_DK2_ERR_STOPPED ： R_DK2_Unload 関数コール、または、R_DK2_Inactivate 関数コールにより、R_DK2_Start 関数で設定したパラメータの転送、または、回路の入出力データの転送が停止したことを表します。

戻り値

なし

解説

R_DK2_Load 関数の引数 pprocess で指定するコールバック関数です。R_DK2_Start 関数で開始した処理の完了を通知します。R_DK2_Unload 関数で強制アンロードなどを行わない場合、本コールバック関数のコール回数は、R_DK2_Start 関数のコール回数と同じになります。
本関数は割込みコンテキストで実行されます。本関数内で、DRP Driver の関数はコールしないでください。

注意

引数 result が R_DK2_ERR_DEVICE のときは、R_DK2_Start 関数で設定したパラメータ、回路の入出力データを配置したメモリの設定などを見直してください。

5.9 R_DK2_GetStatus

R_DK2_GetStatus			DRP Driver API
DRP 上の回路の状態を取得			同期関数
書式	#include "r_dk2_if.h" int32_t R_DK2_GetStatus(const uint8_t id);		
	id	I	状態を取得する回路の ID を指定してください。
戻り値	R_DK2_STATUS_LOADED	:	正常終了。 指定した回路の状態が Loaded state であることを示します。
	R_DK2_STATUS_ACTICATED	:	正常終了。 指定した回路の状態が Activated state であることを示します。
	R_DK2_STATUS_STARTED	:	正常終了。 指定した回路の状態が Started state であることを示します。
	R_DK2_STATUS_LOADING	:	正常終了。 指定した回路の状態が Loading state であることを示します。
	R_DK2_ERR_ARG	:	異常終了。 引数 id が DRP 上の回路ではない場合、本エラーが発生します。
	R_DK2_ERR_OS	:	異常終了。 OS による排他制御に失敗した場合、本エラーが発生します。
解説	本 API 関数は DRP 上の回路の状態を取得します。戻り値が正の数の場合、本関数が成功したことを示し、その値が回路の状態を示します。戻り値が負の数の場合、本関数は失敗したことを示し、その値はエラーコードを表します。DRP 上の回路の状態については「6.2 回路毎の状態遷移」を参照してください。 本 API 関数は、複数の DRP Driver の API 関数が同時に実行されないように OS の機能で排他制御されています。排他制御時の資源獲得がタイムアウトなどで失敗した場合には、戻り値 R_DK2_ERR_OS を返して本関数は失敗します。		
注意	なし		

5.10 R_DK2_GetInfo

R_DK2_GetInfo

DRP Driver API

コンフィグレーションデータの情報を取得、CRC のチェック　　同期関数

書式

```
#include "r_dk2_if.h"
int32_t R_DK2_GetInfo(const void *const pconfig, config_info_t *const pinfo, const bool crc_check);
```

pconfig	I	情報を取得するコンフィグレーションデータのアドレスを指定してください。 コンフィグレーションデータは 32 バイト境界に整列させてください。
pinfo	O	構造体 config_info_t 型の変数のアドレスを指定してください。本 API 関数は、構造体のメンバに以下に示すコンフィグレーションデータの情報を格納します。

メンバ名	型	説明
type	uint8_t	予約領域です。0 が格納されています。
pname	char *	回路名を表す最大 31 バイトの文字列へのポインタが格納されます。
ver	uint32_t	コンフィグレーションデータのバージョンが格納されます。※
cid	uint32_t	コンフィグレーションデータに格納されている回路を表す固有の ID が格納されます。

※　ver の格納形式は、以下のようになります。

ビット位置	説明
0～7	ビルド番号が格納されます。
8～15	マイナーバージョンが格納されます。
16～23	メジャーバージョンが格納されます。
24～31	予約領域です。0 が格納されます。

例えば、ver の値が 0x00010201 の場合、Ver.1.21 であることを表します。

crc_check	I	情報取得時にコンフィグレーションデータの CRC のチェックを行うか否かを真偽値で指定します。

戻り値

R_DK2_SUCCESS	:	正常終了。
R_DK2_ERR_ARG	:	異常終了。 pconfig が NULL、または、pinfo が NULL の場合、本エラーが発生します。
R_DK2_ERR_FORMAT	:	異常終了。 コンフィグレーションデータのフォーマット不整合を検出した場合、本エラーが発生します。
R_DK2_ERR_CRC	:	異常終了。 引数 crc_check に true が指定された、かつ、コンフィグレーションデータの CRC が不正な場合、本エラーが発生します。

解説

本 API 関数は引数 pconfig で指定されたアドレスに配置されたコンフィグレーションデータの情報を取得します。取得したコンフィグレーションデータの情報は引数 pinfo で指定されたアドレスに書き込まれます。
また、本 API 関数はコンフィグレーションデータの CRC チェックを行います。CRC チェックに失敗した場合は、戻り値 R_DK2_ERR_CRC を返却して異常終了します。

注意

戻り値が R_DK2_ERR_FORMAT となった場合は、引数 pconfig で指定したアドレスが、正しいコンフィグレーションデータのアドレスとなっているか確認してください。

5.11 R_DK2_GetVersion

R_DK2_GetVersion	DRP Driver API
DRP Driver のバージョン情報を取得	同期関数

書式

```
#include "r_dk2_if.h"
uint32_t R_DK2_GetVersion(void);
```

戻り値

DRP Driver のバージョン情報　：　格納形式は以下のようになります。

ビット位置	説明
0～7	ビルド番号が格納されます。
8～15	マイナーバージョンが格納されます。
16～23	メジャーバージョンが格納されます。
24～31	予約領域です。0 が格納されます。

例えば、戻り値が 0x00010201 の場合、Ver.1.21 であることを表します。

解説

本 API 関数は DRP Driver のバージョン番号を取得します。

注意

なし

6. 状態遷移

6.1 DRP Driver 全体の状態遷移

図6.1に DRP Driver 全体の状態遷移とクロックの供給について示します。

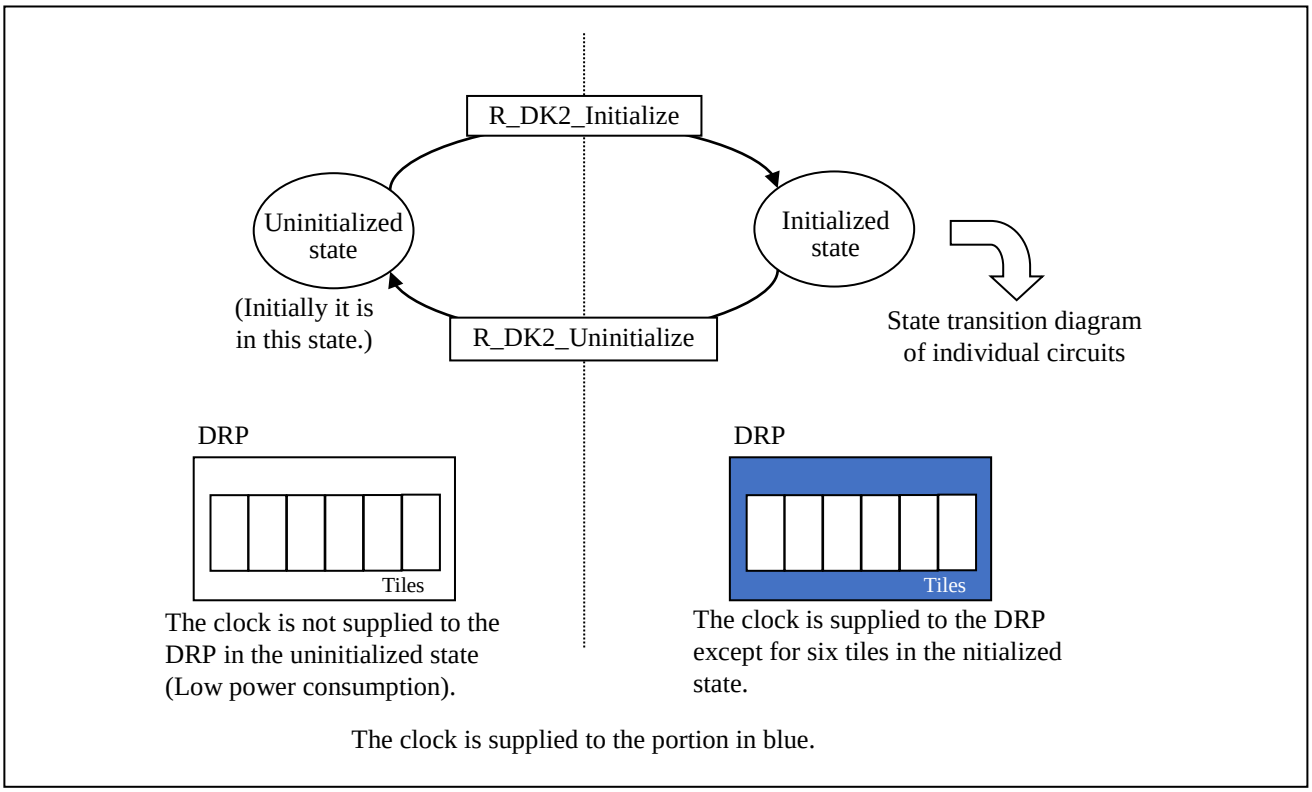


図6.1 DRP Driver 全体の状態遷移とクロック供給について

6.2 回路毎の状態遷移

図6.2に回路ごとの状態遷移とクロック供給について示します。

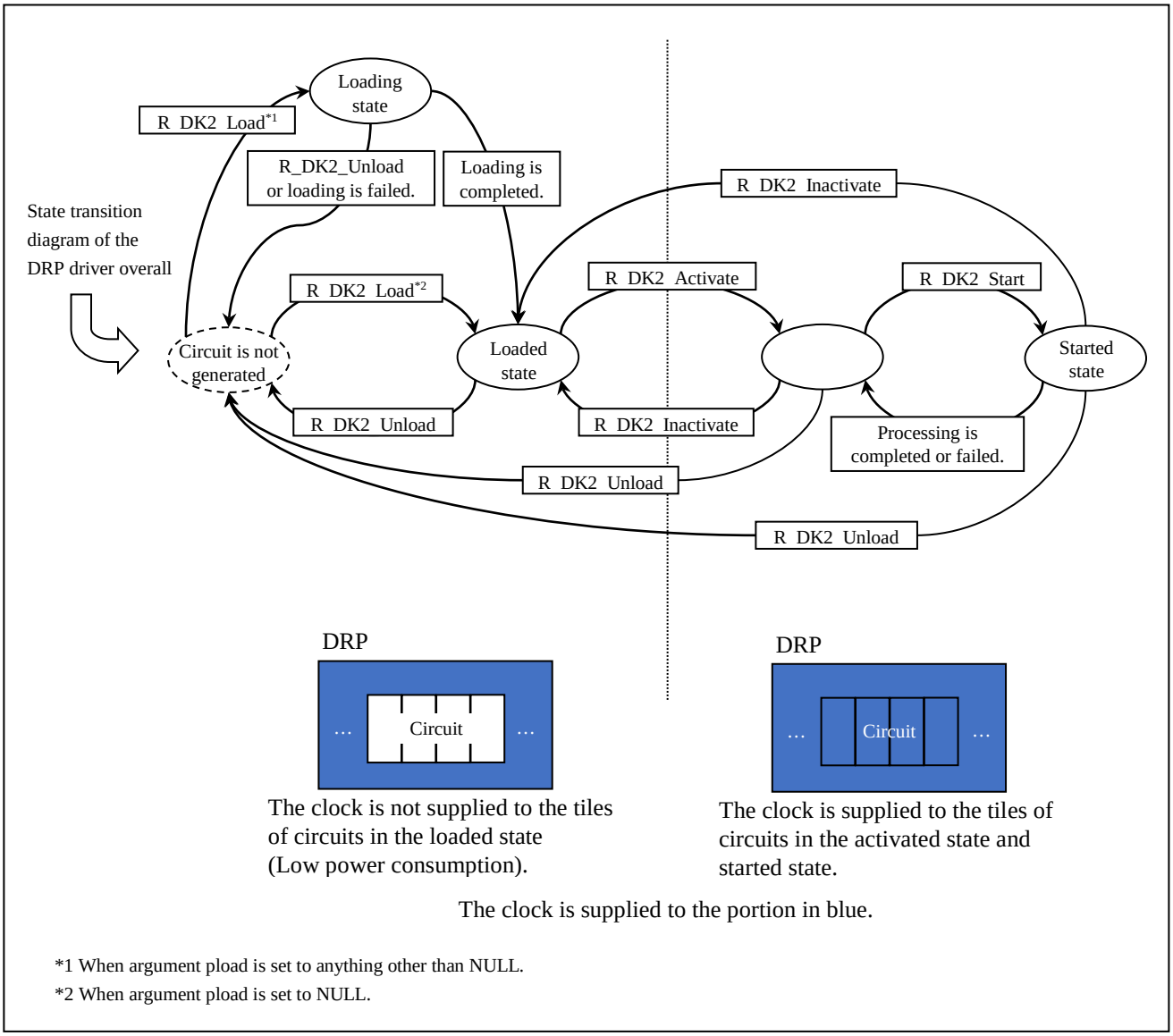


図6.2 回路毎の状態遷移とクロック供給について

7. 制御フローチャート

図7.1に DRP Driver の使い方の例をフローチャートで示します。

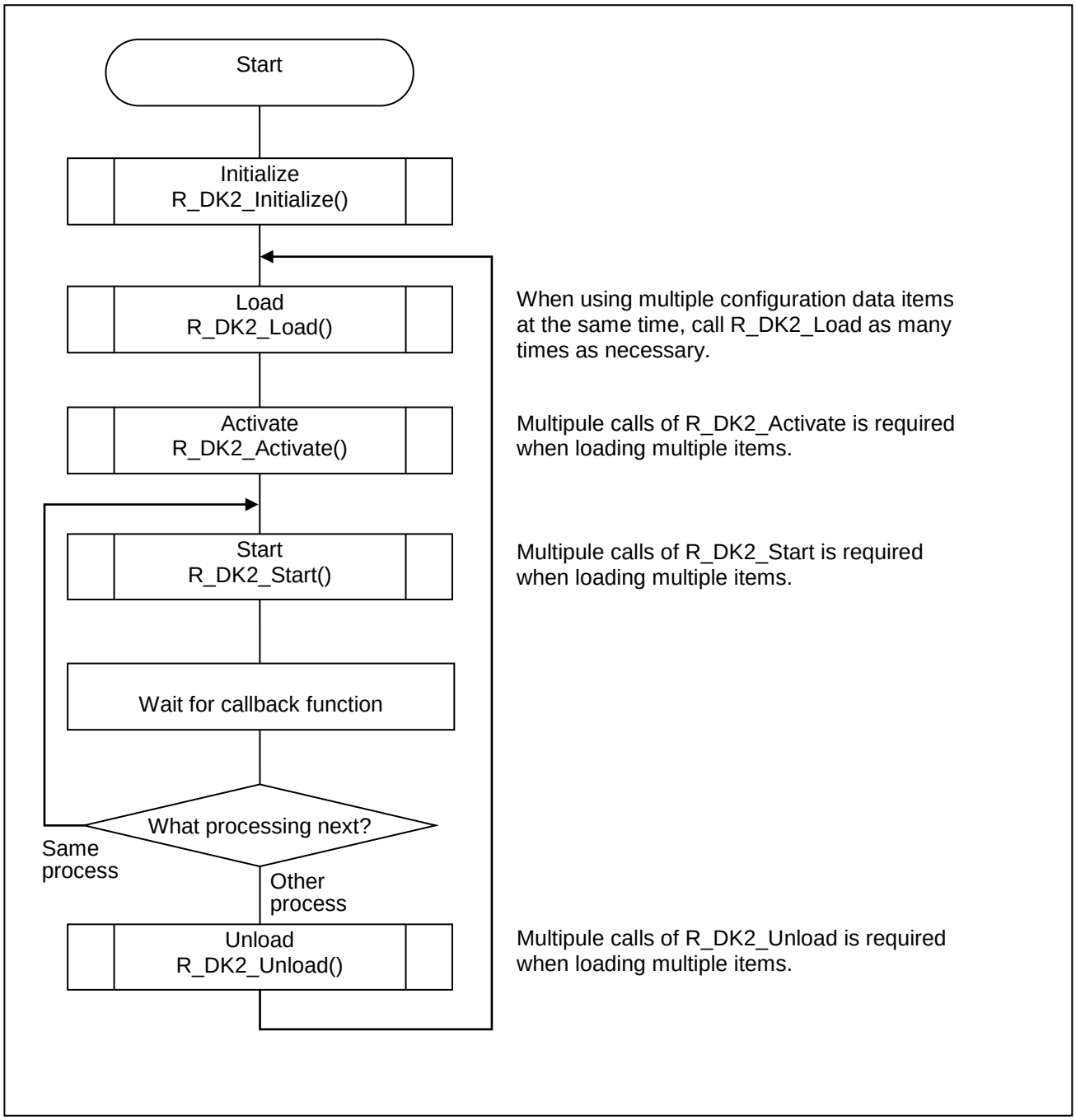


図7.1 DRP Driver の使い方の例

8.　OS 依存部

DRP Driver の OS 依存部は、OS abstraction layer として分離されており、DRP Driver は OS abstraction layer を介して、FreeRTOS をサポートします。

DRP Driver が OS 依存部を用いて提供する機能は、API 関数のリエントラント対応です。FreeRTOS のミューテックスを用いて排他制御を行うことにより、表8.1に記載した一部の API 関数についてリエントラントが可能です。

DRP Driver は、リエントラント性を実現するため、一つのミューテックスを用いて排他制御を行います。リエントラント可能 API 関数実行時に、他のリエントラント可能 API 関数をコールした場合、実行中の API 関数が終了するまでウェイトします。

r_dk2_if.c で定義されたマクロ MUTEX_WAIT を用いて、排他制御時のタイムアウト時間を設定することができます。タイムアウト時間を設定する場合は、マクロ MUTEX_WAIT に 0 から 0xFFFFFFFF までの整数を設定してください。設定値はミリ秒単位のタイムアウト時間を表します。0 はウェイトを行わないことを意味します。デフォルトではタイムアウト時間は 100 ミリ秒に設定されています。

表8.1　　DRP Driver API 関数のリエントラント可能関数一覧

API 関数名	リエントラント対応	Page
R_DK2_Initialize	リエントラント不可	7
R_DK2_Uninitialize	リエントラント不可	8
R_DK2_Load	リエントラント可能	9
R_DK2_Unload	リエントラント可能	14
R_DK2_Activate	リエントラント可能	15
R_DK2_Inactivate	リエントラント可能	16
R_DK2_Start	リエントラント可能	17
R_DK2_GetStatus	リエントラント可能	19
R_DK2_GetInfo	リエントラント不可	20
R_DK2_GetVersion	リエントラント不可	21

9. 関連ドキュメント

ユーザーズマニュアル：ハードウェア

RZ/A2M グループ ユーザーズマニュアル ハードウェア編（R01UH0746）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：ソフトウェア

RZ/A2M グループ DRP Library ユーザーズマニュアル（R01US0367）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

ルネサスエレクトロニクス統合開発環境（e2 studio）に関しては、最新版をルネサスエレクトロニクスホームページから入手してください。

テクニカルアップデート／テクニカルニュース

（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

10. ドライバのインポート方法

10.1　e² studio

Smart Configuratorツールを使用してe² studioのプロジェクトにドライバをインポートする方法の詳細については、RZ/A2M Smart Configuratorユーザーガイド：e² studio R20AN0583JJを参照してください。

10.2　e² studio 以外で作成されたプロジェクトの場合

このセクションでは、ドライバをプロジェクトにインポートする方法について説明します。
一般的に、どのIDEにも2つのステップがあります。

1) プロジェクトに必要なソースツリー内の場所にドライバをコピーします。
2) ドライバをコピーした場所へのリンクをコンパイラに追加します。

他に必要なドライバがある場合（例えばr_cbufferなど）、同様にインポートする必要があります。

改訂記録	RZ/A2M グループ DRP Driver ユーザーズマニュアル

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2018.09.14	ー	初版発行
1.01	2019.05.31	27	「10. ドライバのインポート方法」の章を追加。

RZ/A2M グループ DRP Driver ユーザーズマニュアル

発行年月日　2018年9月14日　Rev.1.00
2019年5月31日　Rev.1.01

発行　ルネサス エレクトロニクス株式会社
〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

RZ/A2M グループ

ルネサス エレクトロニクス株式会社

R01US0355JJ0101

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 3.1